Réingénierie

Quand nous concevons des applications, nous devons chercher à prévoir les évolutions futures, à anticiper les modifications et à créer des classes et des méthodes faiblement couplées et hautement cohésives, mais aussi faciles à modifier. C'est un objectif noble, mais nous ne pouvons bien sûr anticiper toutes les adaptations ni préparer notre code pour toutes les extensions auxquelles nous pouvons penser.

Concept

La **réingénierie** est l'activité de restructuration d'une conception existante pour maintenir une bonne conception des classes quand une application est modifiée ou étendue.

C'est pourquoi la *réingénierie* est importante.

La réingénierie est l'activité de restructuration des classes et méthodes existantes afin de les adapter à des modifications de fonctionnalités et de contraintes. Durant la vie d'une application, nous ajoutons souvent de nouvelles fonctionnalités. Un des effets secondaires est que les classes et les méthodes ont tendance à grossir.

Il est tentant pour un programmeur chargé de la maintenance d'ajouter des lignes de code aux classes ou aux méthodes existantes. Agir de la sorte peut cependant réduire le degré de cohésion. Quand nous ajoutons de plus en plus de code à une méthode ou à une classe, il est vraisemblable que cela représente plus qu'une tâche ou entité précise.

La réingénierie invite à repenser et à « re-concevoir » les structures des méthodes et des classes. Les classes sont souvent séparées en deux, et les méthodes sont divisées en deux ou plus. La réingénierie peut aussi conduire à fusionner plusieurs classes ou méthodes, mais ce cas est plus rare.

Réingénierie et test

Avant de fournir un exemple de réingénierie, nous devons réfléchir au fait que lorsque nous repensons une application, nous pouvons proposer des modifications importantes à quelque chose qui fonctionnait. Et lorsque nous changeons quelque chose, nous risquons d'introduire des erreurs. C'est pourquoi il faut agir avec prudence et nous devons nous assurer de l'existence d'un ensemble de tests pour la version actuelle du programme avant l'étape de réingénierie. Si ces tests n'existent pas, la première étape sera d'en créer plusieurs qui seront utiles pour réaliser un test de non-régression sur la version obtenue par réingénierie. La réingénierie ne doit débuter que lorsque ces tests sont prêts. Idéalement, la réingénierie doit se dérouler en deux étapes :

- La première étape est une réorganisation qui fournit les mêmes fonctionnalités que la version originale. En d'autres termes, nous restructurons le code source pour améliorer sa qualité et non pour modifier ou augmenter ses fonctionnalités.

Une fois cette étape accomplie, les tests de non-régression doivent être passés pour s'assurer que nous n'avons pas introduit d'erreurs.

- La seconde étape intervient seulement quand nous avons retrouvé les principales fonctionnalités dans la version réorganisée. Nous sommes alors en mesure d'étendre le programme dans de bonnes conditions. Une fois le travail terminé, nous devons bien sûr tester la nouvelle version.

Apporter de nombreuses modifications simultanément (réorganisation et ajout de fonctionnalités) rend plus difficile la localisation des problèmes quand ils apparaissent.